University of Turku / Department of Mathematics and Statistics
SCIENTIFIC COMPUTING
Exercise 02 / Solutions

1. A data set $(x_j, y_j), j = 1, ..., m$, and a fixed point (s, t) is given. Consider the lines $y = t + k(x - s)$ through this point and for varying parameter k form the sum

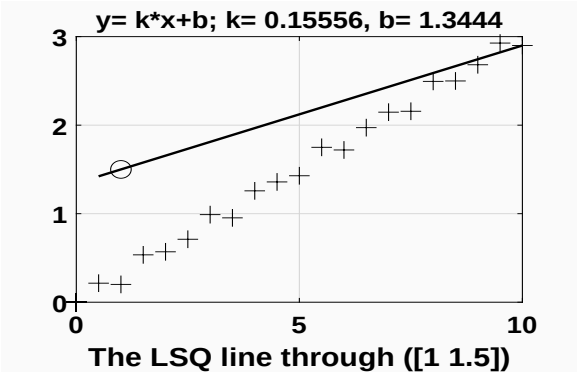
$$A(k) = \sum_{j=1}^{m} (y_j - t - k(x_j - s))^2 .$$

Write the formula for the derivative $A'(k)$, and solve the linear equation $A'(k) = 0$ for k. With this particular value of k we get the least square (LSQ) line fitted to our data set through the point (s, t). Generate some synthetic data [e.g. $x_j = 0.5 * j, y_j = 0.3 * x_j + 0.1 * \sin(30 * x_j), j = 1, ..., 20$,] plot the data, and by the above method, the LSQ-line through the point (s, t) $s = 1, t = 1.5$, and check visually whether the line fits nicely to the data set.

Solution:

```
% FILE d021.m begins.
% A'(k) = 2 \sum_{j=1}^m {- (x_j-s)(y_j-t -k(x_j -s))}
% The requirement A'(k) =0 yields
% k= numer/denom;
% denom= sum((x(j)-s)*(x(j)-s))
% numer= sum((x(j)-s)*(y(j)-t))
% The LSQ-line is y= t+ k*(x-s) = k*x + (t-k*s)
clear
close all
%path(path,'../../util')
%startup
x=zeros(20);
y=zeros(20);
%s=0.5; t=0.7;
s=1.0; t=1.5;
for j=1:20
  x(j)=0.5*j;
%  y(j)= 0.3*x(j)*log(x(j)) + 0.1*sin(30*x(j));
  y(j)= 0.3*x(j) + 0.1*sin(30*x(j));
end
denom= sum((x(j)-s).*(x(j)-s));
numer= sum((x(j)-s).*(y(j)-t));
k= numer/denom;
b=t-k*s;
myx=[x(1) x(20)]; myy=k*myx+b;  % Plot the LSQ line
figure(1)
h=axes;
set(h,'FontWeight','bold','FontSize',20)
plot(myx, myy,'b-','LineWidth', 2)
grid on;
hold on
title(['y= k*x+b; k= ' num2str(k) ', b= ' num2str(b)],...
      'FontSize',20,'FontWeight','bold')
xlabel(['The LSQ line through (' mat2str([s, t]) ')'])
plot(s,t,'ko',x,y,'k+','MarkerSize',15)
% FILE d021.m ends.
```

Output:



2. P. J. Myrberg (1892–1976) has published the following algorithm for the computation of the square root in his paper in Ann. Acad. Sci. Fenn. Ser. A I 253 (1958), 1-19. Fix $z \in \mathbb{C} \setminus \{1\}$ and let $q_0 = (z+1)/(z-1)$, $q_{k+1} = 2q_k^2 - 1, k = 0, 1, ...$ Then $\sqrt{z} = \Pi_{k=0}^{\infty}(1 + 1/q_k)$. Carry out MATLAB tests to check this claim in the following cases

(a) z is real and > 1,
(b) z is real and in $[0, 1]$.
(c) z is complex [Hint: Generate a random complex number w and set $z = w * w$ and see whether the algorithm gives you w.]

Solution:

```
% FILE d022.m begins.
% P.J. Myrberg algorithm
function h022()
for mmax=4:9
      fprintf('mmax= %3d\n',mmax)
      fprintf('(a): z>1: ')
      w= 2:0.02:8;
      z=w.*w;
      w2=sqrtPJM(z,mmax);
      fprintf('  %12.4e\n',norm(w-w2))
      fprintf('(b): 0<z<1: ')
      w= 1./(2:0.02:8);
      z=w.*w;
      w2=sqrtPJM(z,mmax);
      fprintf('  %12.4e\n',norm(w-w2))
      fprintf('(c): z complex : ')
      w= 2:0.02:8; w=w+i*rand(size(w));
      z=w.*w;
      w2=sqrtPJM(z,mmax);
      fprintf('  %12.4e\n',norm(w-w2))
end

function y=sqrtPJM(z,mmax)
q=(z+1)./(z-1);
myqlst=[q];
if (nargin<2)
      mmax=4;
end
for j=1:mmax
q=2*(q.^2) -1;
myqlst=[myqlst; q];
end
tmp=1+1./myqlst;
y=prod(tmp);
% FILE d022.m ends.
```

Output:

```
mmax=   4
(a): z>1:     2.3242e-02
(b): 0<z<1:     4.0246e-04
(c): z complex :     2.4599e-02
mmax=   5
(a): z>1:     5.6100e-06
```

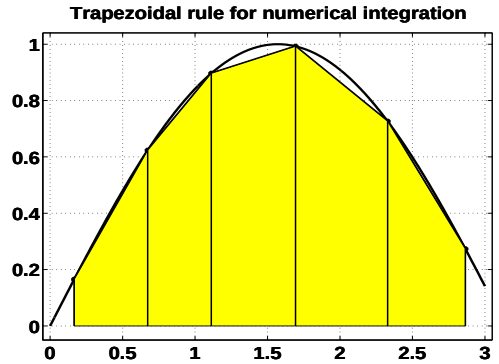


```
(b): 0<z<1:     9.2529e-08
(c): z complex :     6.1816e-06
mmax=   6
(a): z>1:     4.3042e-13
(b): 0<z<1:     7.1504e-15
(c): z complex :     5.1476e-13
mmax=   7
(a): z>1:     7.9690e-14
(b): 0<z<1:     2.5732e-15
(c): z complex :     8.7909e-14
mmax=   8
(a): z>1:     7.9690e-14
(b): 0<z<1:     2.5732e-15
(c): z complex :     9.0254e-14
mmax=   9
(a): z>1:     7.9690e-14
(b): 0<z<1:     2.5732e-15
(c): z complex :     7.1215e-14
```

3. The use of the Trapezoid formula for numerical integration of a tabulated function $f(x_i) = y_i$ is based on the formula

$$s = \sum_{i=1}^{n} (x_{i+1} - x_i)(y_{i+1} + y_i)/2,$$

where the tabulated values are $(x_i, y_i), i = 1, \ldots, n+1$, and $x_i < x_{i+1}$.



(a) Show that if $f(x) = \sum_{j=1}^{m} c_j \sin(d_j * x)$, then by high school calculus

$$\int_a^b f(x)dx = \sum_{j=1}^{m} (c_j/d_j)(\cos(d_j * a) - \cos(d_j * b)).$$

(b) Generate coefficient vectors c and d [e.g. `c=3*rand(1,5); d= 1+10*rand(1,5)`] and use the exact formula in part(a) to compute the integral $\int_0^1 f(x)dx$ and also compute the value by the trapezoid formula and print the error when the points $x_j = (1/n) * j, j = 0, 1, ..., n$, are used in the trapezoid formula with $n = 20 : 10 : 200$.

Solution:

```
% FILE  d023.m  begins.
% USES: d023f.m
clear;
m=5;
c=3*rand(1,m); d=1+9*rand(1,m); % All d(j)'s must be different
% from 0
fprintf('\n  n-1   trapz        exact         difference\n')
for n=21:10:201
a=0; b=1;  dx=(b-a)/(n-1);
x=a:dx:b;
outp=[];
exact= sum((c./d).*cos(a*d)) -sum((c./d).*cos(b*d));
y=d023f(c,d,x);
% vaihtoehtoinen tapa: y=sum( diag(c)*sin((d')*x) );
trapfor=0.5*sum((y(1:n-1)+y(2:n)).*(x(2:n)-x(1:n-1)));
myerr=trapfor-exact;
outp=[outp; n-1  trapfor exact myerr];
fprintf('%4d  %12.4e  %12.4e  %12.4e \n',outp')
end
% FILE  d023.m  ends.

function y=d023f(cvec,dvec,x)
p=length(cvec);
if (p ~= length(dvec))
  error('Incompatible coefficients in d023f');
end;
tmp=0;
for j=1:p
  tmp=tmp+cvec(j)*sin(dvec(j)*x);
end;
y=tmp;
% FILE d023f.m ends.
```

Output:


```
 n-1   trapz        exact         difference
 20    1.8555e+00    1.8647e+00   -9.2117e-03
 30    1.8606e+00    1.8647e+00   -4.0887e-03
 40    1.8624e+00    1.8647e+00   -2.2989e-03
 50    1.8632e+00    1.8647e+00   -1.4710e-03
 60    1.8637e+00    1.8647e+00   -1.0214e-03
 70    1.8639e+00    1.8647e+00   -7.5035e-04
 80    1.8641e+00    1.8647e+00   -5.7446e-04
 90    1.8642e+00    1.8647e+00   -4.5388e-04
100    1.8643e+00    1.8647e+00   -3.6764e-04
110    1.8644e+00    1.8647e+00   -3.0383e-04
120    1.8644e+00    1.8647e+00   -2.5530e-04
130    1.8645e+00    1.8647e+00   -2.1753e-04
140    1.8645e+00    1.8647e+00   -1.8756e-04
150    1.8645e+00    1.8647e+00   -1.6339e-04
160    1.8645e+00    1.8647e+00   -1.4360e-04
170    1.8646e+00    1.8647e+00   -1.2720e-04
180    1.8646e+00    1.8647e+00   -1.1346e-04
190    1.8646e+00    1.8647e+00   -1.0183e-04
200    1.8646e+00    1.8647e+00   -9.1903e-05
```

4. For random points a, b, c, d in the plane, find the point of intersection of the lines L_1 through a, b and L_2 through c, d. Plot the picture of the lines and the point of intersection.

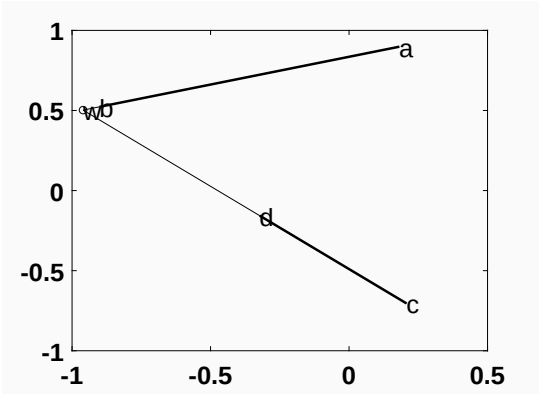
Solution:

```
% FILE d027.m begins.
function q=d027
close all
clear
a=rand+i*rand;
b=-rand+i*rand;
c=rand-i*rand;
d=-rand-i*rand;
% Method I
u=(a-b).*(c.*conj(d)- conj(c).*d)-(c-d).*(a.*conj(b)- conj(a).*b);
v=conj(a-b).*(c-d)- conj(c-d).*(a-b);
w=u./v;
x=real([a b w]);
y=imag([a b w]);
plot(x,y,'b-')
hold on
x2=real([a b ]);
```

```
y2=imag([a b ]);
plot(x2,y2,'k-','LineWidth',2)
x1=real([c d w]);
y1=imag([c  d  w]);
plot(x1,y1,'r-',real(w),imag(w),'ko')
x3=real([c d ]);
y3=imag([c  d ]);
plot(x3,y3,'k-','LineWidth',2)
text(real(a),imag(a),'a','FontSize',20)
text(real(b),imag(b),'b','FontSize',20)
text(real(c),imag(c),'c','FontSize',20)
text(real(d),imag(d),'d','FontSize',20)
text(real(w),imag(w),'w','FontSize',20)
print -dps d027.ps
%Method II
sol=intsec(a,b,c,d)
w1=a+ sol(1)*(b-a);
w2= c+ sol(2)*(d-c);
% w, w1, w2 should all be equal:
fprintf('abs(w-w1) =%12.4e, abs(w-w2)=  %12.4e  \n',abs(w-w1),abs(w-w2));


function sol=intsec(w1,x,w2,y)
% FIND t and s such that
% w1+ t(x-w1)=  w2+ s(y-w2)
mymat=[ real(x-w1) -real(y-w2);   imag(x-w1) -imag(y-w2) ];
myrhs=[real(w2-w1); imag(w2-w1)];
sol=mymat\myrhs;
%w3= w1+ sol(1)*(x-w1);
%w4= w2+ sol(2)*(y-w2);
%fprintf('If everything is OK, this should be zero : %12.4e\n',abs(w3-w4))
% FILE d027.m ends.
```

Output:

```
sol =

    1.0553
    2.2028

abs(w-w1) =  1.5701e-16, abs(w-w2)=    2.4825e-16
```

5. Fix $a > 0$. It is well-known that if $0 < x_0 < 2/a$ the recursive sequence defined by $x_{k+1} = x_k(2 - ax_k)$ converges to $1/a$.

(a) Write a MATLAB function to compute the reciprocal with this method.

(b) Apply the method to several test cases (say to $a = 0.1 * k, k = 1 : 1000$,) and report the error.

Solution:

```
% FILE d025.m begins.
% Plots the log of the abs of mean error
% USES: d025f.m, which uses myrecip.m
close all
clear all
global a   % needed in d025f.m
global x   % needed in d025f.m
for k=1:1000
    a(k)=0.1*k;
end
fprintf('   n   maximum error\n')
jrec=1:10;   %the iterations
for j=1:jrec(end)
    x=0.5*ones(size(a))./a;   %the starting point
    tmp=d025f(j);
    error(j)=tmp(1);
    maxerror(j)=tmp(2);
    fprintf('  %2d   %8.4e\n',j,maxerror(j));
end
axes('FontSize',[18],'FontWeight','bold');
```

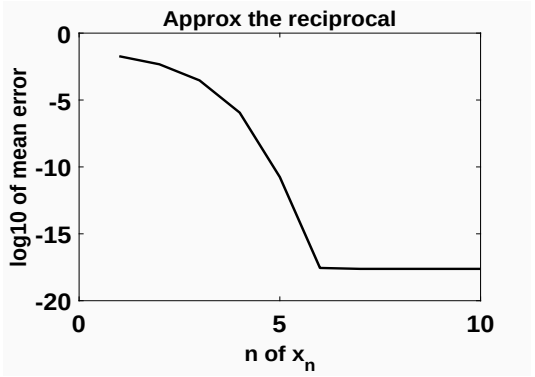
```
grid on
plot(jrec,log10(error),'LineWidth',2)
title('Approx the reciprocal','FontSize',[18],'FontWeight','bold')
      xlabel('n of x_n','FontSize',[18],'FontWeight','bold')
      ylabel('log10 of mean error', 'FontSize',[18],'FontWeight','bold')
% FILE d025.m ends.

% FILE d025f.m begins.
%calculates the mean error for
%approximation of the reciprocal with x .
%                                      j
function z=d025f(jrec)
global a
y=myrecip(a,jrec);
y2=1./a;
z=[mean(abs(y-y2)) max(abs(y-y2))];

% FILE d025f.m ends.

% FILE myrecip.m begins.
%the function for ex. 5 (a)-part
%based on the convergence:
%
%  x   = x (2 - a x ) , lim x  = 1/a,  when   0 < x < 2/a.
%   k+1   k        k     k   k                    0
function y= myrecip(a,mmax)
global x
for j=1:mmax
x=x.*(2-a.*x);
end
y=x;
% FILE myrecip.m ends.
```

Output:

```
n   maximum error
1   2.5000e+00
2   6.2500e-01
3   3.9063e-02
4   1.5259e-04
5   2.3283e-09
6   4.4409e-16
7   4.4409e-16
8   4.4409e-16
```



```
9   4.4409e-16
10  4.4409e-16
```



6. Numerical analysis textbooks often point out that the quadratic formula $x_{1,2} = (-b \pm \sqrt{b^2 - 4ac})/(2a)$ for the solution of the equation $ax^2 + bx + c = 0$ may lead to distorted results when $|4ac|$ is very small. Plan a MATLAB experiment to justify this remark and report the errors in the test cases you have used. (Hint. Choose $a = b^2/(4c) \cdot 10^{-m}$, $m = 3, \ldots, 15$. Compare your result either to what you get from the method explained during the lectures or from MATLAB `roots([a b c])`.)

Solution:

```
% FILE d026.m begins.
% xj are from the ordinary quadratic formula, r from the roots -function
% and yj from the formula (see Numerical Recipes in C, section 5.6 p. 184)
%                          ----------
%  q = - (1/2)( b + sign(b)\/ b^2 - 4ac ),  y_1  = q/a   and  y_2  = c/q .
%
function h026
fprintf(...
' m     ero        |r(2)-x2|       r(2)        x2        y2\n')
for m=3:15
    abc=rand(1,3);
    a=abc(1); b=abc(2); c=abc(3); a= b*b*10^(-m)/(4*c);
    abc(1)=a;
    [x1, x2]=myquadformula(abc);
    [y1, y2]=thequadformula(abc);
    r=roots(abc);
```

```
    fprintf('%2d.  %8.4e  %8.4e %10.4f %10.4f %10.4f\n', ...
    m,norm([x1 x2]'- roots(abc)),norm(r(2)-x2),r(2),x2,y2)
end
function [x1, x2]=myquadformula(abc)
    a=abc(1); b=abc(2); c= abc(3);
    s=b*b-4*a*c;
    x1=[(-b -sqrt(s))/(2*a)];
    x2=[ (-b +sqrt(s))/(2*a)];
function [y1, y2]=thequadformula(abc)
    a=abc(1); b=abc(2); c=abc(3);
    s=b*b-4*a*c;
    q=-0.5*(b+sign(b)*sqrt(s));
    y1=q/a;
    y2=c/q;
% FILE d026.m ends.
```

Output:

```
 m     ero        |r(2)-x2|       r(2)        x2        y2
 3.  2.5846e-13  2.5846e-13    -5.6315    -5.6315    -5.6315
 4.  1.0008e-12  1.0008e-12    -1.2676    -1.2676    -1.2676
 5.  5.8483e-11  5.6728e-12    -0.7029    -0.7029    -0.7029
 6.  1.9085e-09  4.1566e-10    -3.2625    -3.2625    -3.2625
 7.  7.4809e-09  6.7316e-10    -1.3310    -1.3310    -1.3310
 8.  7.6273e-09  7.6273e-09    -0.9377    -0.9377    -0.9377
 9.  3.9416e-06  9.9200e-07    -5.4302    -5.4302    -5.4302
10.  1.2251e-04  1.0386e-05   -20.6325   -20.6325   -20.6325
11.  3.0985e-05  5.3624e-06    -0.5572    -0.5573    -0.5572
12.  1.6101e-04  1.6101e-04    -1.2842    -1.2840    -1.2842
13.  6.4005e-02  1.3797e-02    -8.1725    -8.1863    -8.1725
14.  3.1381e-02  2.8640e-03    -0.5863    -0.5892    -0.5863
15.  2.7767e-02  2.7767e-02    -0.3113    -0.2835    -0.3113
```